University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
S. Faour, M. Fuchs, A. Mályusz

Algorithm Theory
Exercise Sheet 7

Due: Friday, 6th of December, 2024, 10:00 am

Exercise 1: Worst Case Decrease *(4 Points)*

We've seen in the lecture that Fibonacci heaps are only efficient in an *amortized* sense. However, the time to execute a single, individual operation can be large. Show that in the worst case, the `decrease-key` operation can require time $\Omega(n)$ (for any heap size n).

Hint: Describe an execution in which there is a decrease-key operation that requires linear time.

Exercise 2: Fibonacci Heaps Modifications - Amortized I *(5 Points)*

Suppose we "simplify" Fibonacci heaps such that we do *not* mark any nodes that have lost a child and consequentially also do *not* cut marked parents of a node that needs to be cut out due to a `decrease-key`-operation. Is the *amortized* running time

(a) ... of the `decrease-key`-operation still $\mathcal{O}(1)$? *(1 Point)*

(b) ... of the `delete-min`-operation still $\mathcal{O}(\log n)$? *(4 Points)*
Hint: Can we still guarantee the recursive property (proved in the lecture) i.e. a given node with rank i has i children that have at least ranks $i-2, i-3$, ..., respectively?

Explain your answers.

Exercise 3: Fibonacci Heaps Modifications - Amortized II *(11 Points)*

(a) Assume that operation `decrease-key` never occurs. Show that in this case, the maximum rank $D(n)$ of a Fibonacci heap is at most $\lfloor \log_2(n) \rfloor$. *(4 Points)*

(b) We want to augment the Fibonacci heap data structure by adding an operation `increase-key`(v, k) to increase the key of a node v (given by a direct pointer) to the value k. The operation should have an amortized running time of $\mathcal{O}(\log n)$. Describe the operation `increase-key`(v, k) in sufficient detail and prove the correctness and amortized running time. *(7 Points)*

Remark: You can use the same potential function as for the standard Fibonacci heap data structure. Note however that after conducting `increase-key`(v, k) the Fibonacci heap must still be a list of heaps, where the maximum rank $D(n) \in O(\log n)$.